# Detecting Domain-specific Events based on Robot Sensor Data

Bernhard G. Humm[1] and Guglielmo van der Meer[2]
[1]*Hochschule Darmstadt, University of Applied Sciences, Darmstadt, Germany*
[2]*KUKA Roboter GmbH, Augsburg, Germany*

Keywords: Robots, Time Series, Complex Event Detection.

Abstract: This paper presents an approach for detecting domain-specific events based on robot sensor data. Events may be error situations as well as successfully executed manufacturing steps, depending on the application domain at hand. The approach includes segmenting streams of sensor data into meaningful intervals and subsequently matching patterns on those segments. Pattern matching is performed in near real-time allowing events to be detected continuously during the execution of a robotics application. The approach is demonstrated by means of a real-world manufacturing use case, namely the automated assembly of electrical components by a robot. The approach has been implemented prototypically had has been evaluated successfully.

## 1 INTRODUCTION

The role of robotics is becoming increasingly important in various application domains, particularly in industrial manufacturing (Wang et al., 2018). Recently developed robots are sensitive, i.e., they integrate sensors like joint torque sensors which allow measuring physical values like forces on the robot. Sensitive robots allow for new application scenarios, e.g., collaborative robotics in which a robot and can share its workspace with humans (Fantini et al., 2018). In this paper, we examine another application scenario for sensitive robotics, namely detecting events from robot sensor data.

Complex event processing (Cugola and Margara, 2015) is a method of tracking and analysing streams of data about events, and deriving conclusions from them. Events are relevant occurrences that happen in a certain application domain. In industrial manufacturing, events may be error situations as well as successfully executed manufacturing steps, depending on the application domain at hand.

In this paper we present an approach for detecting domain-specific events based on robot sensor data. This approach includes segmenting streams of sensor data into meaningful intervals and subsequently matching patterns on those segments, where the patterns are given in a *symbolic* form.

The remainder of this paper is structured as follows. In Section 2, we introduce a sample application use case in order to motivate and demonstrate our approach. Section 3 presents related work. In Section 4, we specify the problem statement by means of requirements. Section 5 is the core of this paper presenting our approach. Section 6 briefly describes a sample implementation. Section 7 evaluates our approach. Section 8 concludes our paper and indicates future work.

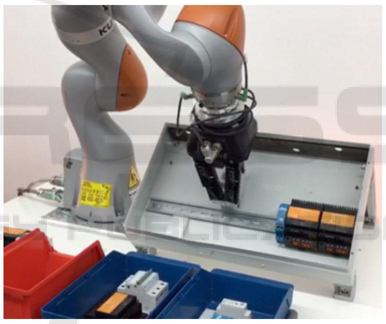
Figure 1: Automated electrical component assembly by robot.

## 2 SAMPLE USE CASE

We motivate our approach by means of a sample application use case, namely the automated assembly of electrical components by a robot. See Figure 1.

In this application use case, a sensitive robot (in this case a KUKA LBR iiwa 7) picks an electrical component, e.g. a fuse, from a container and mounts it onto a profile rail within a switch cabinet. The

Humm, B. and van der Meer, G.
Detecting Domain-specific Events based on Robot Sensor Data.
DOI: 10.5220/0007952503980405
In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, pages 398-405
ISBN: 978-989-758-380-3

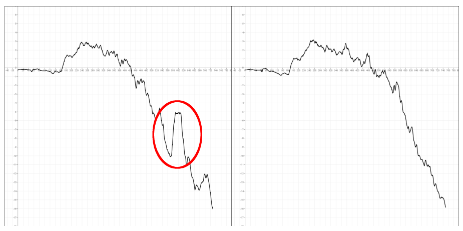
Figure 2: Comparison of force signals at (a) successful assembly and (b) unsuccessful assembly.

assembly is successful if the electrical component snapped into the profile rail. However, various situations can cause an unsuccessful assembly. Examples are defects in the snapping mechanism of the electrical component, obstructions on the profile rail, or slight deviations in the positioning of the rail. In such cases, the electrical component will not be assembled successfully but may fall down.

Detecting unsuccessful assembly is of utmost importance to the assembly process since subsequent manufacturing steps rely on the success of previous steps. In this paper, we focus on event detection based on the assembling robot's sensor data.

Figure 2 shows the time series of the sensor signals of a typical successful and unsuccessful assembly step.

The figure shows the force in dimension z, i.e., in the direction of the flange mounting the electrical component. On successful assembly, a characteristic curve shape can be observed, as marked with a red circle in Figure 2 (a). This curve characteristic is caused by the electrical component snapping in: A sharp increase in force followed by a sudden decrease. In situations of unsuccessful assembly, e.g. due to a defect in the snapping mechanisms, this curve characteristic can usually not be observed.

The approach presented in this paper allows discriminating between successful and unsuccessful assembly by analysing the time series of the force sensor signals.

## 3 RELATED WORK

There is a variety of sources for time series data, including financial data (stock prices), electroencephalograms (EEGs), climate- and weather data (temperatures, wind speeds), human motion data, as well as sensor data in industrial applications. Therefore, the task of detecting patterns in time series data has been addressed by many researches, focussing on different application domains. The survey by (Fakhrazari and Vakilzadian, 2017) categorizes the data mining tasks for time series data into indexing, clustering, classification, prediction, anomaly detection, motif discovery, rule discovery, summarization and visualization.

In this paper, we focus on the *classification* of time series data. For a given time series, we want to associate it with a certain class, namely the class of the *event* that caused a certain shape of the time series.

The main building blocks for the efficient detection of events in streaming time series data are the preprocessing of the incoming raw data, the resulting representation, and the actual pattern detection that is applied to this representation..

### 3.1 Preprocessing

The goals of preprocessing time series data usually are to generate a compressed or simplified representation of the time series in order to allow indexing and querying time series databases, to increase the efficiency of subsequent data mining and analysis processes in general.

In the context of pattern detection or pattern search, the compressed representations of time series data are usually based on variuos forms of *segmentation*. For example, in order to support pattern searches in time series data, (Pratt and Fink, 2002) suggests a segmentation that is based on the detection of ”important points”, specifically minima and maxima, in order to obtain a compressed representation of the time series. Similarly, (Fu et al., 2007) describe a way to detect patterns in stock data based on the identification of perceptually important points (PIP) in the time series. The more general notion of ”change points” as being points where some characteristics of time series data change has been examined by (Aminikhanghahi and Cook, 2017), including a detailed comparison and classification of approaches for preprocessing time series in order to detect these change points.

The survey of (Keogh et al., 1993) classifies approaches for computing segmentations – specifically, piecewise linear approximations – including the aspect of how the approximating line segments are computed. This can be done via interpolation, creating a line connecting the first and the last entry of a time series interval, or via regression, where the approximation is a line that has the lowest least squares error.

The challenges that are imposed by processing streaming time series data in resource-constrained environments has been addressed by (Soroush et al., 2008). They propose an algorithm that has linear time complexity, constant space complexity, and generates the minimum number of line segments that approximate the given data for a given error bound.

For our application case, the primary goal is to obtain a simple representation that captures the overall shape of the incoming signal. Therefore, we use a simple segmentation based on important points, which are essentially minima and maxima, similar to the approach of (Pratt and Fink, 2002).

## 3.2 Representations

Different representations of time series data have been developed, and a taxonomy of time series representations was given in (Lin et al., 2003). A comparison of the performance of many different time series representations and associated distance measures was performed in (Wang et al., 2013).

The result of the preprocessing step in our application case is a segmented time series. There are different options for the actual representation of such a segmented time series. In some cases, like (Pratt and Fink, 2002), the representation may be given by the line segments between important points, augmented with additional information that allows more efficient indexing and queries. In other cases, the segmented time series data is transferred into a *symbolic* representations. With SAX (Symbolic Aggregate approXimation) (Lin et al., 2003), each segment is aggregated and represented by an element of an alphabet.

The most simple representation is that of a piecewise linear approximations of the original time series. For our use-case, this representation is the most suitable one, due to the high compression that can be achieved compared to the original time series data, and the simple, intuitive notion of shapes and patterns that can be expressed with a piecewise linear approximation.

## 3.3 Pattern Matching

The task of detecting various forms of *patterns* in time series data has been addressed by many researches in different application domains. An important distinction for the approaches that have been proposed refers to the particular data mining task.

In some application domains, the goal is to find time series that are similar to a given time series, which is often referred to as *query-by-content* ((Faloutsos et al., 1994)). In other cases, the goal is to find time series that have certain characteristics inside a large time series database (see, for example, (Pratt and Fink, 2002)). In both cases, it is possible to generate an index that allows efficient queries for similar time series. Based on such a database and indexing structures, high-level query languages can be defined. For example, SQL-TS by (Sadri et al., 2001) is a query language that allows `SELECT-FROM-WHERE` queries that are tailored for the properties of time series data.

Indexing structures and classical similarity-based queries cannot be applied when the pattern should be detected in *streaming* time series data. For example, (Wu et al., 2004) describe a query language for event-driven subsequence similarity search based on financial data streams.

The theoretical background for a general query evaulation model for event streams was laid out by (Agrawal et al., 2008). They define the $NFA^b$, a non-deterministic finite automaton with a buffer, that allows describing the pattern matching process as a sequence of state transitions based on the incoming events, and analyze the expressibility of such an automaton.

One of the goals of our approach is to define a simple, intuitive and explainable definition of the patterns, in a symbolic form. It should be possible to express the pattern directly based on the characteristics of the time series representation. Therefore, we chose to represent the patterns using simple conditions that describe these characteristics, which can directly be expressed using any language that supports basic relational operators and conjunctions.

## 4 PROBLEM STATEMENT

We specify the goals of our approach by means of requirements.

**(R1) Event Detection:** The approach shall allow detecting application-specific events based on robot sensor data.

Here, events are relevant situations during the execution of a robotics application, e.g., errors during a manufacturing step. A concrete event detection mechanism is specific to the application domain. The approach shall allow specifying criteria for domain-specific events.

**(R2) Explainable:** The criteria for domain-specific events can be specified by domain experts and explained to and discussed with other domain experts. Depending on the criticality of the robotics application, this may be an important aspect of quality assurance.

In the sample application use case introduced above, snapping an electronic component into a profile rail results in a characteristic shape of the curve force in z dimension caused by the snapping mechanism: A sharp increase in force followed by a sudden decrease. This curve characteristic can be interpreted by an engineer as a successful snap-in event.

**(R3) Continuous Detection:** Events shall be detected *during* the execution of the robotics application, not ex post.

This means that the detection is performed continuously on the incoming sensor data. How much delay between the occurrence and the detection of an event is acceptable, is domain-specific. For example, in the use case introduced above, a delay of 1s could be deemed acceptable.

**(R4) Accurate:** The classification performance of the detector shall be appropriately high.

Various measurements of classification performance may be employed, e.g., accuracy or F1 score. Which measurement and which classification performance is suitable is domain-specific. For example, in the use case introduced above, an accuracy of 0.85 could be deemed necessary.

**(R5) Feasible:** The approach shall be implementable in a real-world robotics manufacturing setting. This takes into consideration limited computing power of a robot and limited bandwidth between robot and data analytics server due to network and protocol limitations.

## 5 APPROACH

### 5.1 Overview

Figure 3 shows the conceptual view on the approach presented in this paper.

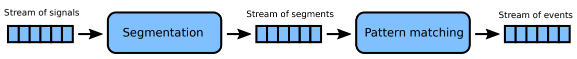

Figure 3: The conceptual view on the proposed approach.

A stream of low-level signals is obtained from the robot sensors. This stream is preprocessed to compute the segments, which are an approximation of the raw data. The segments are then passed to the pattern matcher. When a sequence of segments matches the desired pattern, a high-level event is generated.

The following sections summarize the definitions that serve as the basis for our approach. The next sections explain the main building blocks of our approach, which are the *segmentation* and the *pattern matching* step.

### 5.2 Definitions

The basic definitions given here refer to the elements of Figure 3, and consist of a formal definition of the signal stream, segments, and patterns.

#### 5.2.1 Signal Stream

The stream of low-level signals that is obtained from the robot sensor can be described as numerical time series. Such a time series $S$ is a sequence of entries $s_{i\in\mathbb{N}_0}$, where $s_i = (t_i, v_i)$, with $t_i$ being the *time stamp*, and $v_i$ being the *value* of the signal at index $i$. The time stamps are assumed to be numeric and strictly increasing, meaning that $t_i < t_{i+1}$, and thus, to define an order on the entries.

#### 5.2.2 Segments and Segment Measures

We refer to a *segment* as a line segment, which, in the context of this paper, is an approximation of a connected subset of the entries of a time series. A segment therefore is a tuple $I = ((t_{\text{start}}, v_{\text{start}}), (t_{\text{end}}, v_{\text{end}}))$ where $(t_{\text{start}}, v_{\text{start}})$ are a time stamp and value of the start of the segment, and $(t_{\text{end}}, v_{\text{end}})$ the time stamp and value of the end of the segment, with $t_{\text{start}} < t_{\text{end}}$.

A sequence of segments that are computed from a time series of a signal can thus be written as $(I_i)_{i\in\mathbb{N}_0}$. For an interval $[a,b)$ of the natural numbers, $(I_i)_{i\in[a,b)}$ refers to a subsequence of segments that starts at index $a$ and ends before index $b$, thus having a length of $b-a$.

Different *measures* for segments can be defined. For a signal that is represented with a time series $S = (t_i, v_i)_{i\in\mathbb{N}_0}$ and a given segment $I = ((t_{\text{start}}, v_{\text{start}}), (t_{\text{end}}, v_{\text{end}}))$ that has been computed from such a signal, we define

$$\begin{aligned} \Delta_t(I) &= t_{\text{end}} - t_{\text{start}} \\ \Delta_v(I) &= v_{\text{end}} - v_{\text{start}} \\ \text{Slope}(I) &= \Delta_v(I)/\Delta_t(I) \end{aligned} \tag{1}$$

#### 5.2.3 Patterns and Pattern Matches

A *pattern* is a description of the characteristics of a time series which correspond to the occurrence of a domain-specific event. It is a conjunction of *conditions* that say whether a certain relation holds for a given segment. Each condition is therefore an unary predicate $C$ on a segment that compares one ot the segment measures that have been given in Section 5.2.2 to a real number $x \in \mathbb{R}$. Thus, a condition $C$ for a segment $I = ((t_{\text{start}}, v_{\text{start}}), (t_{\text{end}}, v_{\text{end}}))$ has the following form:

$$C(I) = \left\{ \begin{array}{l} t_{\text{start}} \\ t_{\text{end}} \\ v_{\text{start}} \\ v_{\text{end}} \\ \Delta_t(I) \\ \Delta_v(I) \\ \text{Slope}(I) \end{array} \right\} \left\{ \begin{array}{c} > \\ = \\ < \end{array} \right\} x \qquad (2)$$

where $x$ is the threshold value that the segment measure is compared against.

For example, for segments $I_0$ and $I_1$, such conditions may be

$$\begin{aligned} C_0(I_0) = {} & \Delta_v(I_0) > 0.2 \\ C_1(I_0) = {} & \Delta_t(I_0) = 0.5 \\ C_2(I_1) = {} & \text{Slope}(I_1) < 0.3 \end{aligned}$$

We then can define a pattern $P$ as an $m$-ary predicate on segments that is a conjunction of $q$ such conditions: $P = \bigwedge_{j=0}^{j<q} C_j$.

For a sequence of segments $(I_i)_{i\in[a,b)}$ that is the argument list of $P$, the index of the argument for condition $C_j$ is given as $r_j \in [a,b)$. Therefore, a pattern $P$ matches a given sequence $(A_k)_{k\in[a,b)}$ of segments when

$$P((A_k)_{k\in[a,b)}) = \bigwedge_{j=0}^{j<q} C_j(A_{r_j}) \qquad (3)$$

For example, for a pattern $P = C_0 \wedge C_1 \wedge C_2$ based on the conditions $C_0, C_1, C_2$ shown above, we have $r_0 = 0$, $r_1 = 0$, and $r_2 = 1$, and the pattern matches the sequence of segments $(I_0, I_1)$ when

$$\begin{aligned} P(I_0, I_1) = {} & C_0(I_0) \wedge C_1(I_0) \wedge C_2(I_1) \\ = {} & \Delta_v(I_0) > 0.2 \ \wedge \\ & \Delta_t(I_0) = 0.5 \ \wedge \\ & \text{Slope}(I_1) < 0.3 \end{aligned}$$

When a pattern matches a given sequence of segments, the match can be written as a tuple $M = (P, (I_k)_{k\in K})$ with $P$ being the pattern and $(I_k)_{k\in K}$ being the sequence of segments for which the match was detected.

## 5.3 Segmentation

One goal of this paper is to execute the simplification and compression of the input data and the subsequent pattern matching continually, in order to detect the desired events during the operation. This means that we have to compute a highly compressed, simplified representation of the time series data *on the fly*. Therefore, we focus on *piecewise linear approximations* (PLA) for time series.

### 5.3.1 Preprocessing

The raw input data that is obtained from the sensors may contain noise. There are different possible preprocessing steps for the raw data that can reduce the amount of noise.

An overview and comparison of different noise reduction methods for time series is given in (Köhler and Lorenz, 2005). Since we are operating on streaming data, approaches that are based on frequency analysis are not directly applicable. So in order to meet the requirement imposed by the goal of continually processing the data stream, we reduce the noise by applying a simple moving average (SMA) to the raw input data. The SMA is a convolution of the streaming time series data with a rectangular pulse, and thus computes the unweighted mean of a certain number of previous data points. Therefore, it can easily and efficiently be implemented even on robot hardware with limited memory and computing power.

The choice of a SMA for denoising is justified in our application case by the fact that the magnitude of the noise in the input signal is small, and its frequency is high compared to the overall shape that has to be detected. For application cases where the input signal contains other forms of noise, more sophisticated noise reduction methods, like the ones compared in (Köhler and Lorenz, 2005), can be used.

The smoothed time series data is then passed to the actual segmentation routine.

### 5.3.2 Segment Computation

For the main use case addressed in our paper, we can apply a simple segmentation approach: We track the *derivative* of the signal, and emit a new segment whenever the sign of the derivative changes. Algorithm 1 shows how the stream of segments is computed by the stream of preprocessed signals.

### 5.3.3 Segmentation Results

The result of the segmentation process is a continuous stream of segments that is computed from the sensor data: A new segment is emitted whenever the sign of the derivative of the signal changes. A special case that has to be considered is that the signal remains constant for an indeterminate amount of time – for example, when the robot comes to a halt and the recorded forces or velocities do not change any more. To handle this, the segment computation can be extended to emit an interim segment for the pattern matcher when the difference of the time stamp of a new signal and the end of the previously emitted segment is longer than an application specific limit.

```
Input: A stream of signals $s_{i\in\mathbb{N}_0}$, where
       $s_i = (t_i, v_i)$
Output: A stream of segments $I_{j\in\mathbb{N}_0}$
Data: The signal $s_{\text{start}} = (t_{\text{start}}, v_{\text{start}})$
      where the current segment starts,
      initialized with $s_{\text{start}} = s_0$
Data: The derivative $d_{\text{start}}$ of the start of
      the current segment, initialized
      with $d_{\text{start}} = (v_1 - v_0)/(t_1 - t_0)$
for each new signal $s_i$ with $i > 1$ do
    // Compute the derivative at the
      current point
    $d = (v_i - v_{i-1})/(t_i - t_{i-1})$
    if $sign(d) \neq sign(d_{start})$ then
        // Emit a new segment
        $I_j = (s_{\text{start}}, s_{i-1})$
        $d_{\text{start}} = d$
        $s_{\text{start}} = s_i$
        emit $A_j$
    end
end
```

Algorithm 1: Derivative-based segmentation.

```
Input: Pattern $P$, Stream of segments
       $(I_i)_{i\in\mathbb{N}_0}$
Output: Stream of matches $(M_j)_{j\in\mathbb{N}_0}$
for each new segment $I_i$ do
    if $i \geq m$ then
        // Check if the pattern matches
        // the $m$ most recent segments
        $K = [i - m, i + 1)$
        if $P((I_k)_{k\in K})$ then
            $M_j = (P, (I_k)_{k\in K})$
            emit $M_j$
        end
    end
end
```

Algorithm 2: Pattern matching.

### 5.4 Pattern Matching

The goal of the pattern matching process is to detect whether a sequence of segments contains a subsequence that describes the characteristic shape that indicates the occurrence of the domain-specific event that is described via a pattern.

#### 5.4.1 Pattern Matching Process

Algorithm 2 shows the process of the pattern matching. It receives the stream of segments that has been computed by Algorithm 1, and generates a stream of pattern matches.

## 6 IMPLEMENTATION

In order to assess the feasibility of the approaches presented in this paper, we implemented different segmentation algorithms for an actual robot.

Specifically, the prototypical implementations of segmentation algorithms can be executed on the controller of a KUKA LBR iiwa 7 and run synchronously with the control cycle loop at 1kHz. The resulting segment data are passed during the execution of the robotics application to an analytics server for further processing, which includes the pattern detection described in the previous section.

The comparisons and classification benchmarks presented in the following section have been performed with an offline implementation in Java that reads the data from files that contain recordings of the actual robot data.

## 7 EVALUATION

We evaluate our approach by analysing the requirements specified in Section 4.

**(R1) Event Detection:** Our approach presented allows specifying a domain-specific pattern which, when matching on a stream of robot sensor data, emits an event. Here, events are relevant situations during the execution of a robotics application, e.g., errors during a manufacturing step.

**(R2) Explainable:** We consider the pattern language introduced above as explainable to domain experts. The sample pattern can be constructed by a domain expert and can be explained to another expert as being characteristic for a successful snap-in event.

However, to verify this requirement, analyses with many domain experts in various application use cases are required. This is subject to future work.

**(R3) Continuous Detection:** Events are detected continuously *during* the execution of the robotics application, not ex post.

The computing time of the segmentation and pattern matching algorithms is negligible. Even on limited computing resources, the detection logic executes within milliseconds. On a standard desktop PC with 3

GHz CPU, our prototypical implementation in Java 8 took less than 1 millisecond for the segmentation, and less than 1 millisecond for the pattern matching when a new segment was emitted.

**(R4) Accurate:** Which measurement and which classification performance is suitable is domain-specific. In the sample use case presented above, an accuracy of 0.85 for the detection of a successful mounting could be deemed necessary. We have performed an evaluation of our approach based on three different data sets.

Each data set contains 60 recordings of the sensor outputs of the robot that have been measured during the assembly process. For each data set, there are 40 recordings where the mounting process succeeded and 20 recordings where the mounting process failed. The relevant sensor is the sensor that records the force in z-direction, measured at a 1 ms interval. The time series for the data sets $A$, $B$ and $C$ have been smoothed with a simple moving average, with a window size of 25, 35 and 30 respectively.

For all data sets, the same pattern was used:

$$\begin{aligned} P(I_0, I_1, I_2) &= \text{Slope}(I_0) < -0.03 \ \wedge \\ &\quad \text{Slope}(I_1) > 0.03 \ \wedge \\ &\quad \text{Slope}(I_2) < -0.03 \end{aligned}$$

The pattern captures the intuitive description of the required shape of the characteristic element of a curve where the mounting process succeeded: A segment where the force decreased, followed by a sudden increase and a sudden decrease. The threshold values of 0.03 have been chosen by a domain expert, based on knowledge about the magnitude of the change of the force that is to be expected.

Table 1 summarizes the classification results for these test cases.

Table 1: Classification results for the three data sets. The table shows the number of true positives, false positives, true negatives, and false negatives, and the resulting accuracy.

| Data set | TP | FP | TN | FN | acc. |
|---|---|---|---|---|---|
| A | 38 | 0 | 20 | 2 | 0.967 |
| B | 39 | 0 | 20 | 1 | 0.983 |
| C | 34 | 6 | 18 | 2 | 0.867 |

So in this sample use case, the accuracy requirement can be considered as being met.

**(R5) Feasible:** An important aspect of our approach is the strict separation of segment detection and pattern matching. Segment detection operates on large volumes of sensor data, e.g., hundreds of sensor values each millisecond. Transferring such volumes of data from the robot to a data analytics server is often not feasible in a real-world manufacturing setting, due to restrictions of bandwidth and protocols like OPC UA.

Segmentation can be seen as a mechanism of data compression: characteristic information is extracted from the stream of data and the data volume is massively reduced. Table 2 shows the compression that is achieved with the segmentation.

Table 2: The compression that is achieved by the segmentation process. The table shows the average number of time series entries (E) for the 60 recordings of each data set, the window size (WS) of the simple moving average that was used, and the average number of segments (S) that have been generated.

| Data set | avg. E | WS | avg. S |
|---|---|---|---|
| A | 676 | 25 | 21 |
| B | 960 | 35 | 24 |
| C | 963 | 30 | 26 |

The separation of segment detection and pattern matching allows for various deployment options for implementations of our approach. While segmentation and pattern matching may both be executed on the robot server, it can also be split: Segmentation on the robot server and transmission of segment data via protocols like OPC UA to a data analytics server where pattern matching and complex event processing may be executed.

## 8 CONCLUSIONS AND FUTURE WORK

In this paper we have presented an approach for detecting domain-specific events based on robot sensor data. Detection is performed continuously, allowing to react to events during the execution of a robotics application. The approach is feasible and can be implemented in a real-world robotic manufacturing setting. We have implemented the approach prototypically and validated it successfully by means of a sample use case.

We see a great potential in this approach since sensitive robots are increasingly becoming commonplace in industrial manufacturing. Towards this end, we plan to evaluate our approach in a number of application use cases and settings. This includes the application of the approach to detect events that are indicated by different patterns, and a qualitative comparison of the capabilities of detecting complex events in comparison to other approaches and based on different data sets.

An extension of our approach may consider specifying inter-relations between pattens in different time series, e.g., while the speed of the robot flange is in a certain limit, a specific pattern on force in z dimension matches. This requires extending the pattern language with temporal relationships as in (Allen, 1983).

We envisage a large potential in applying machine learning and optimization techniques for learning patterns based on labelled time series. This would allow to largely reduce the time and effort involved in specifying patterns. The fact that patterns are explainable allows domain experts to inspect and validate patterns learned, similarly to decision trees generated by machine learning.

We will continue to publish our results on this.

## ACKNOWLEDGEMENTS

This work was done within the project ProDok 4.0, funded by the German Ministry of Education and Research (BMBF) within the framework of the Services 2010 action plan under funding no. 02K14A110. Executive steering committee is the Karlsruher Institut für Technologie - Karlsruhe Institute of Technology (KIT). Project partners are KUKA Deutschland GmbH, ISRA VISION AG, dictaJet Ingenieurgesellschaft mbH and Hochschule Darmstadt - University of Applied Sciences.

## REFERENCES

Agrawal, J., Diao, Y., Gyllstrom, D., and Immerman, N. (2008). Efficient pattern matching over event streams. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 147–160, New York, NY, USA. ACM.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367.

Cugola, G. and Margara, A. (2015). The complex event processing paradigm. In Colace, F., de Santo, M., Moscato, V., Picariello, A., Schreiber, F. A., and Tanca, L., editors, *Data Management in Pervasive Systems*, Data-Centric Systems and Applications, pages 113–133. Springer, Cham.

Fakhrazari, A. and Vakilzadian, H. (2017). A survey on time series data mining. In *2017 IEEE International Conference on Electro Information Technology, EIT 2017*, pages 476–481. IEEE Computer Society.

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429.

Fantini, P., Pinzone, M., Sella, F., and Taisch, M. (2018). Collaborative robots and new product introduction: Capturing and transferring human expert knowledge to the operators. In Trzcielinski, S., editor, *Advances in Ergonomics of Manufacturing: Managing the Enterprise of the Future*, pages 259–268, Cham. Springer International Publishing.

Fu, T.-C., Chung, K. F.-L., Luk, R. W. P., and man Ng, C. (2007). Stock time series pattern matching: Template-based vs. rule-based approaches. *Eng. Appl. of AI*, 20:347–364.

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (1993). Segmenting time series: A survey and novel approach. In *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company.

Köhler, T. and Lorenz, D. (2005). A comparison of denoising methods for one dimensional time series.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, New York, NY, USA. ACM.

Pratt, K. B. and Fink, E. (2002). Search for patterns in compressed time series. *Int. J. Image Graphics*, 2(1):89–106.

Sadri, R., Zaniolo, C., Zarkesh, A. M., and Adibi, J. (2001). A sequential pattern query language for supporting instant data mining for e-services. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 653–656, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Soroush, E., Wu, K., and Pei, J. (2008). Fast and quality-guaranteed data streaming in resource-constrained sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pages 391–400, New York, NY, USA. ACM.

Wang, T.-M., Tao, Y., and Liu, H. (2018). Current researches and future development trend of intelligent robot: A review. *International Journal of Automation and Computing*, 15(5):525–546.

Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., and Keogh, E. (2013). Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309.

Wu, H., Salzberg, B., and Zhang, D. (2004). Online event-driven subsequence matching over financial data streams. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 23–34, New York, NY, USA. ACM.